Лекция 10

БЫСТРАЯ РАЗРАБОТКА ПРОТОТИПОВ ИНТЕРФЕЙСОВ С ИСПОЛЬЗОВАНИЕМ QT DESIGNER

Лекция 10. Быстрая разработка прототипов интерфейсов с использованием Qt Designer

План

1. Обзор Qt Designer.
2. Создание проекта с GUI в Qt.
3. Редактирование формы в Qt Designer.

1. Обзор Qt Designer

Для **быстрой разработки приложений** (**RAD** – Rapid Application Development) с использованием библиотеки Qt используется специальная программа **Qt Designer**. Ее можно использовать как отдельно, так и совместно с **Qt Creator**.

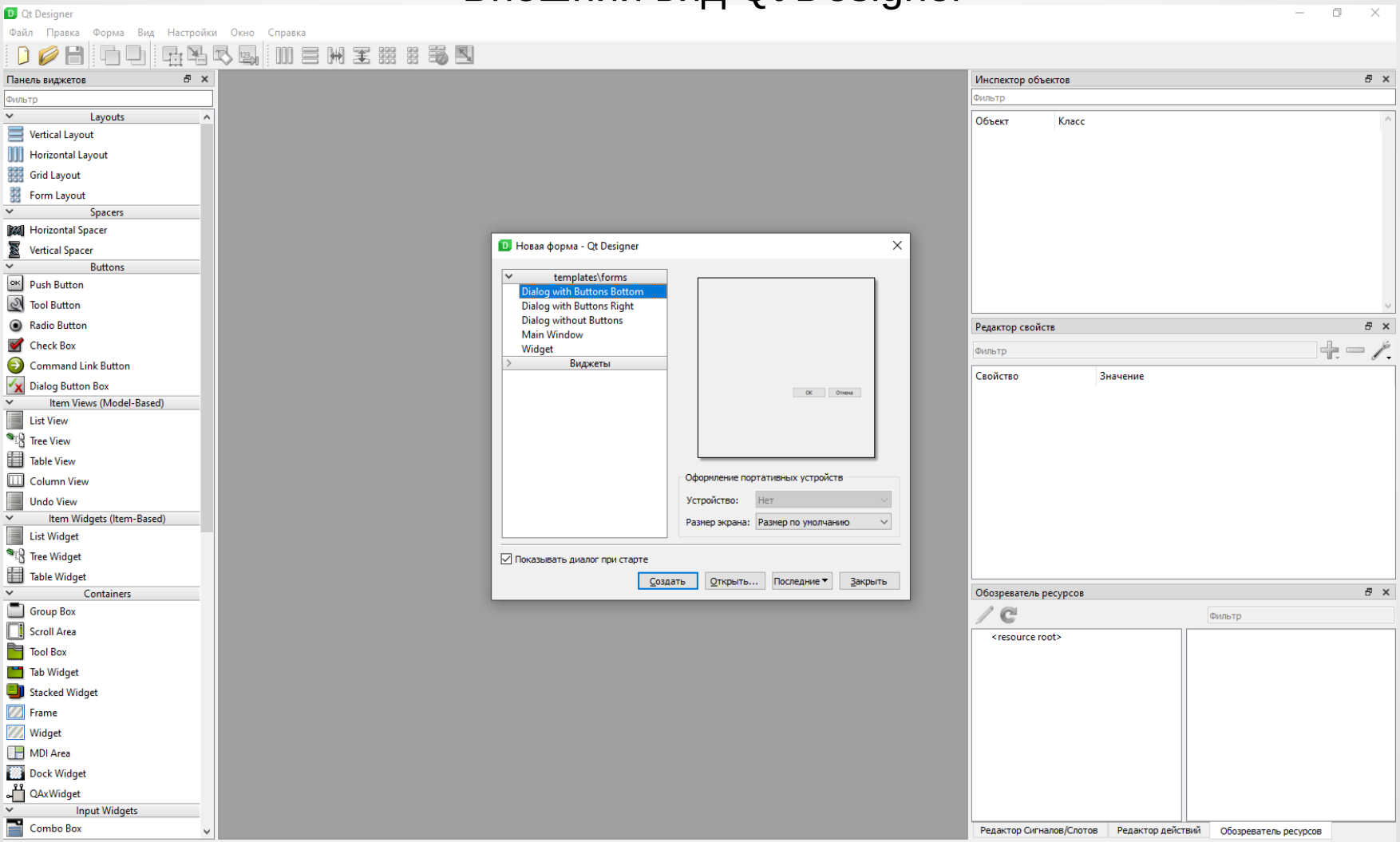
Эта программа предназначена для дизайна интерфейса программы. Принцип ее работы соответствует **WYSIWYG** (What You See Is What You Get), что можно перевести как «**что видишь, то и получишь**».

Qt Designer дает возможность быстро создавать прототипы приложений, базирующихся на **главных окнах**, имеющих **меню**, **строку состояния**, **полосы прокрутки**, **панель инструментов** и т. п.

Созданные в программе Qt Designer файлы описания интерфейса (в формате **XML**) можно конвертировать в исходный код на языке C++.

1. Обзор Qt Designer

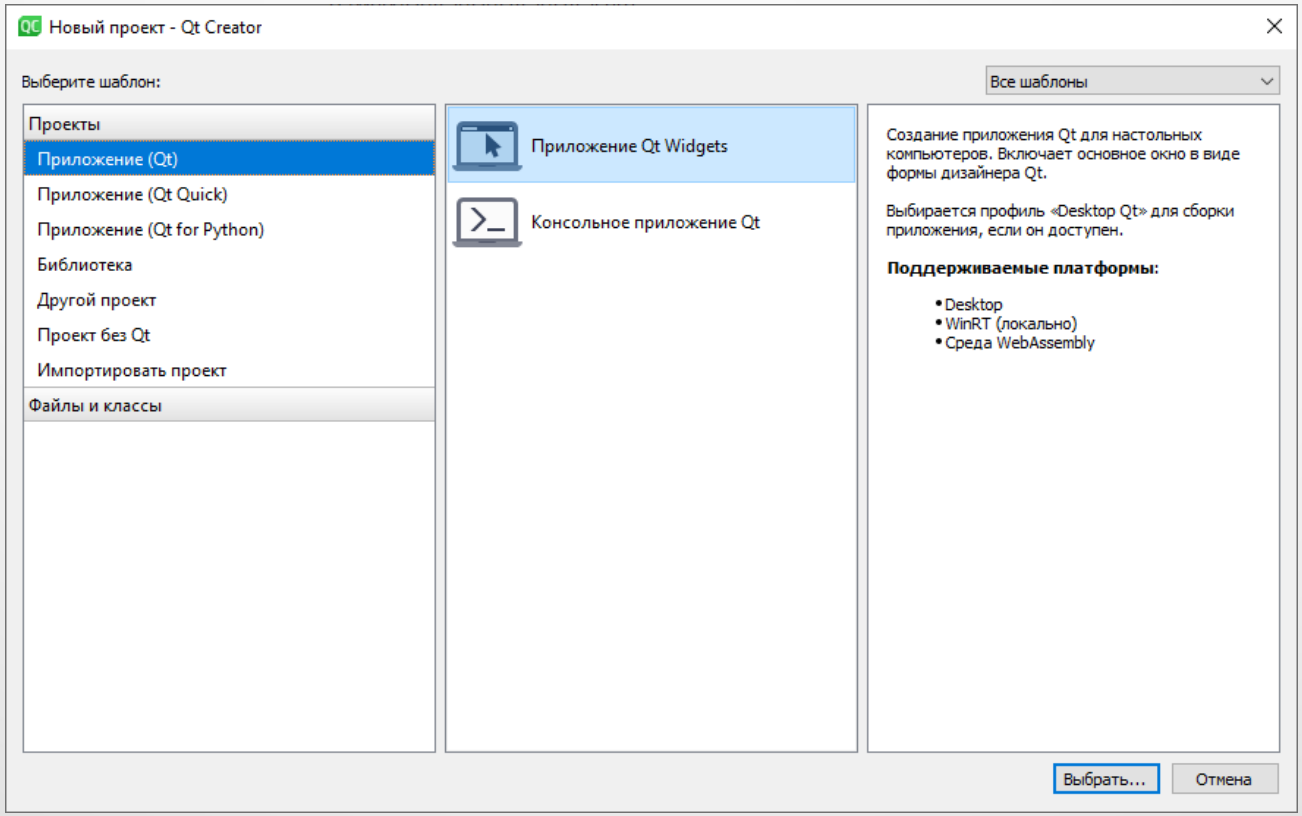
Внешний вид Qt Designer



2. Создание проекта с GUI в Qt

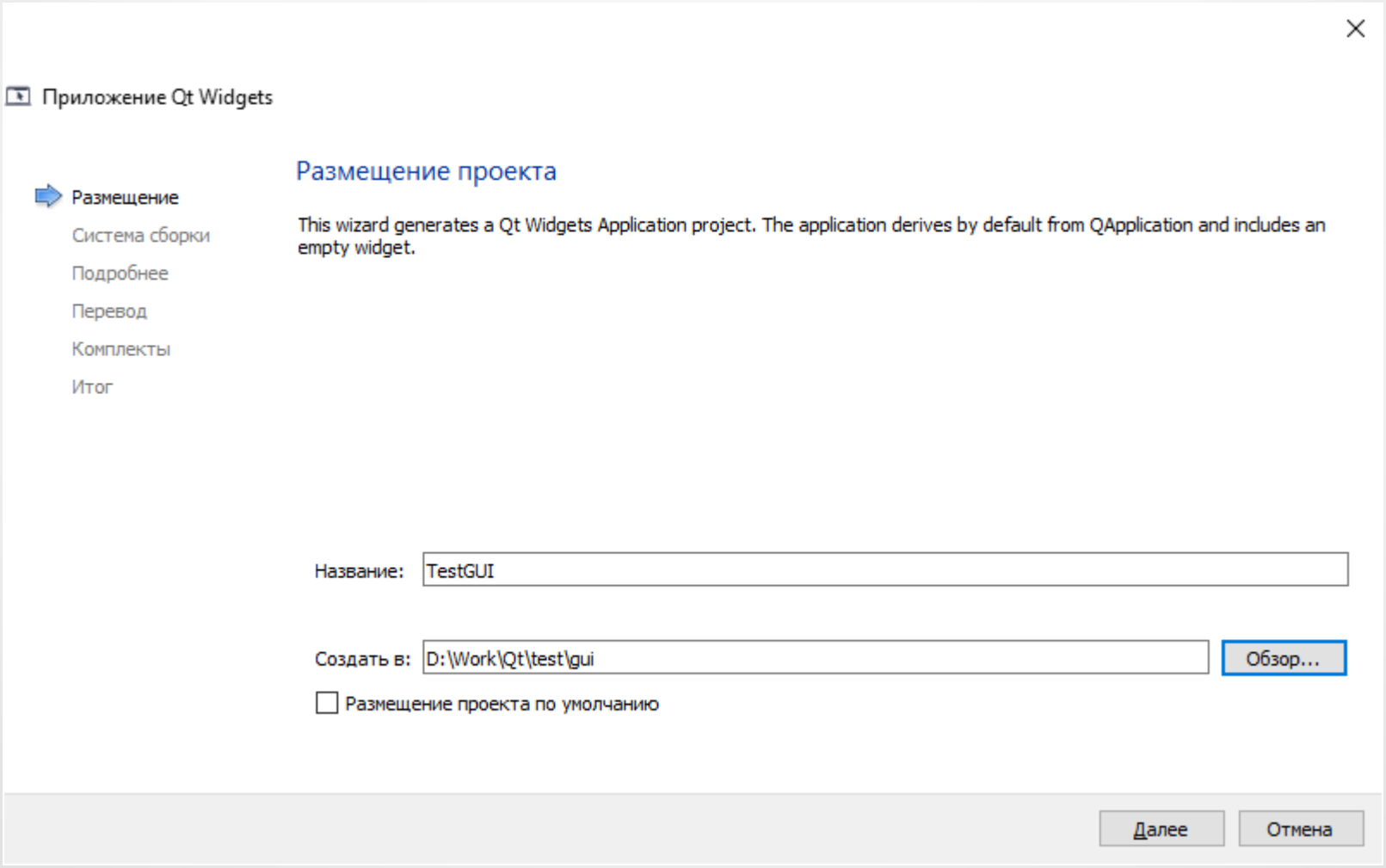
Быстрым способом создания проекта с графическим интерфейсом пользователя в Qt Creator является следующая последовательность действий:

1) выполнить команду главного меню «Файл | Создать файл или проект...»;
2) выбрать шаблон «Приложение (Qt)» => «Приложение Qt Widgets»;



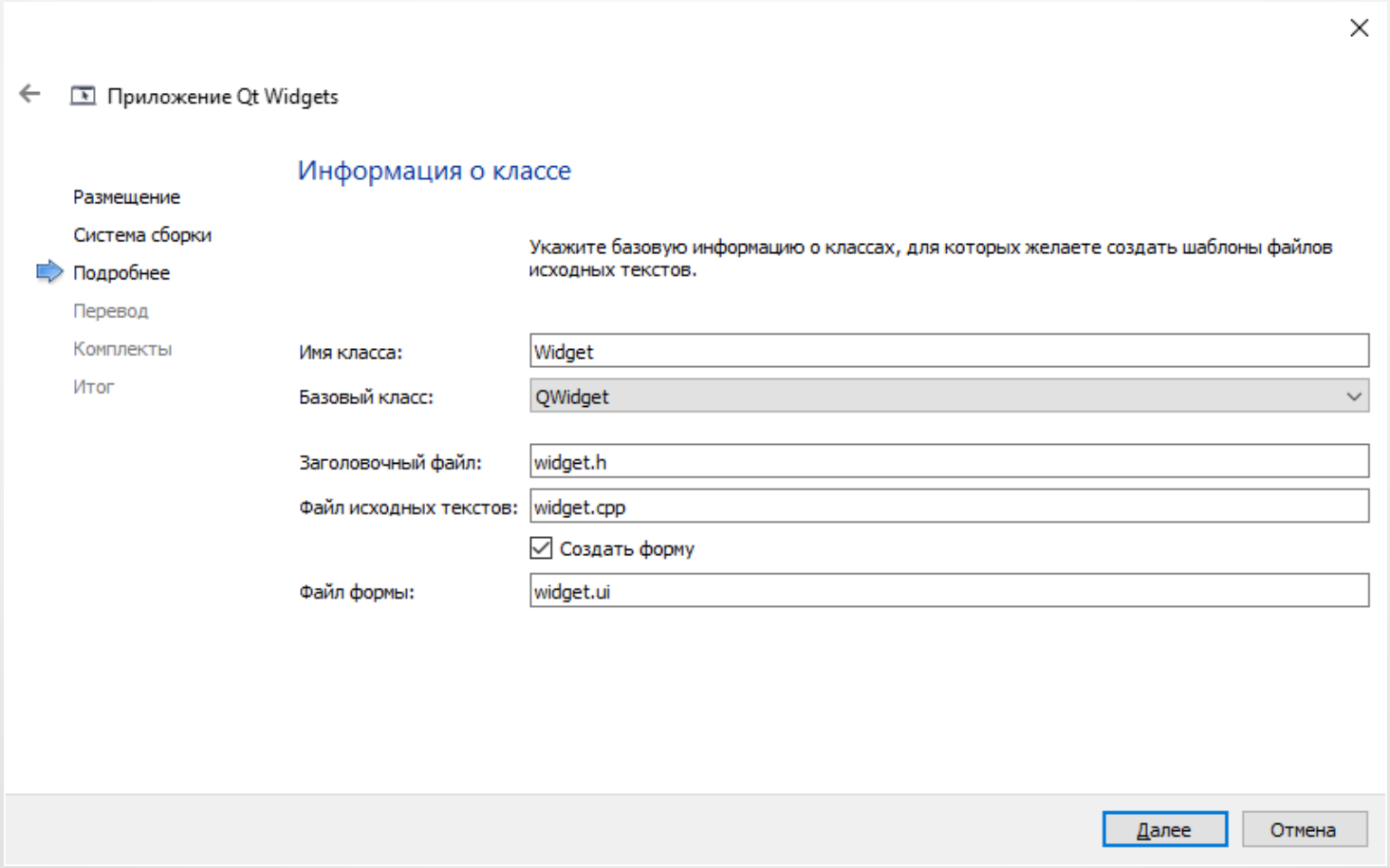
2. Простейшее приложение с GUI в Qt

3) указать название проекта и его месторасположение;



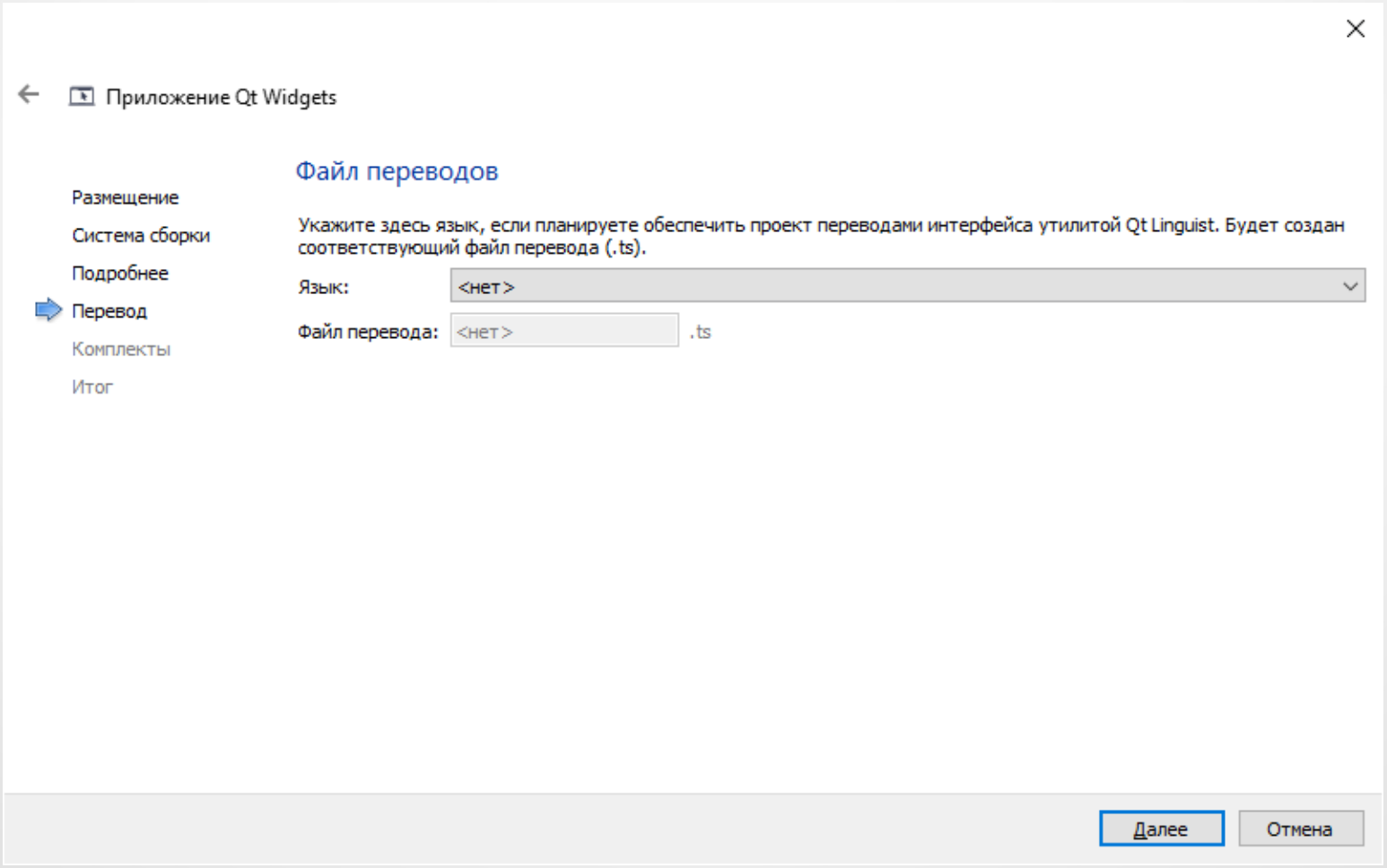
2. Простейшее приложение с GUI в Qt

4) выбрать систему сборки (оптимально – **qmake**);
5) задать информацию о базовом классе, на основании которого будет строится интерфейс главного окна будущей программы;



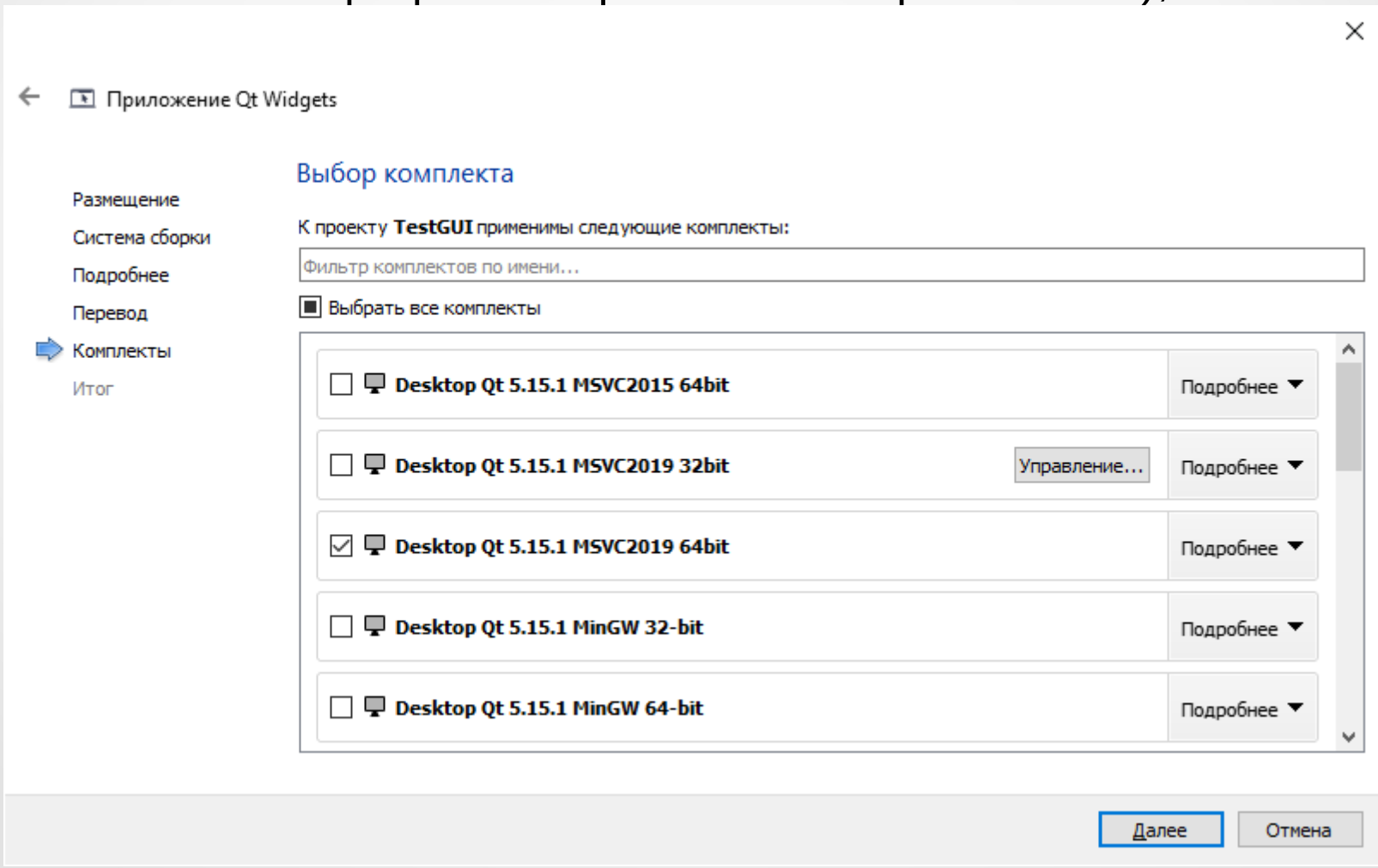
2. Простейшее приложение с GUI в Qt

6) выбрать при необходимости файл переводов, содержащий разные языки интерфейса программы;



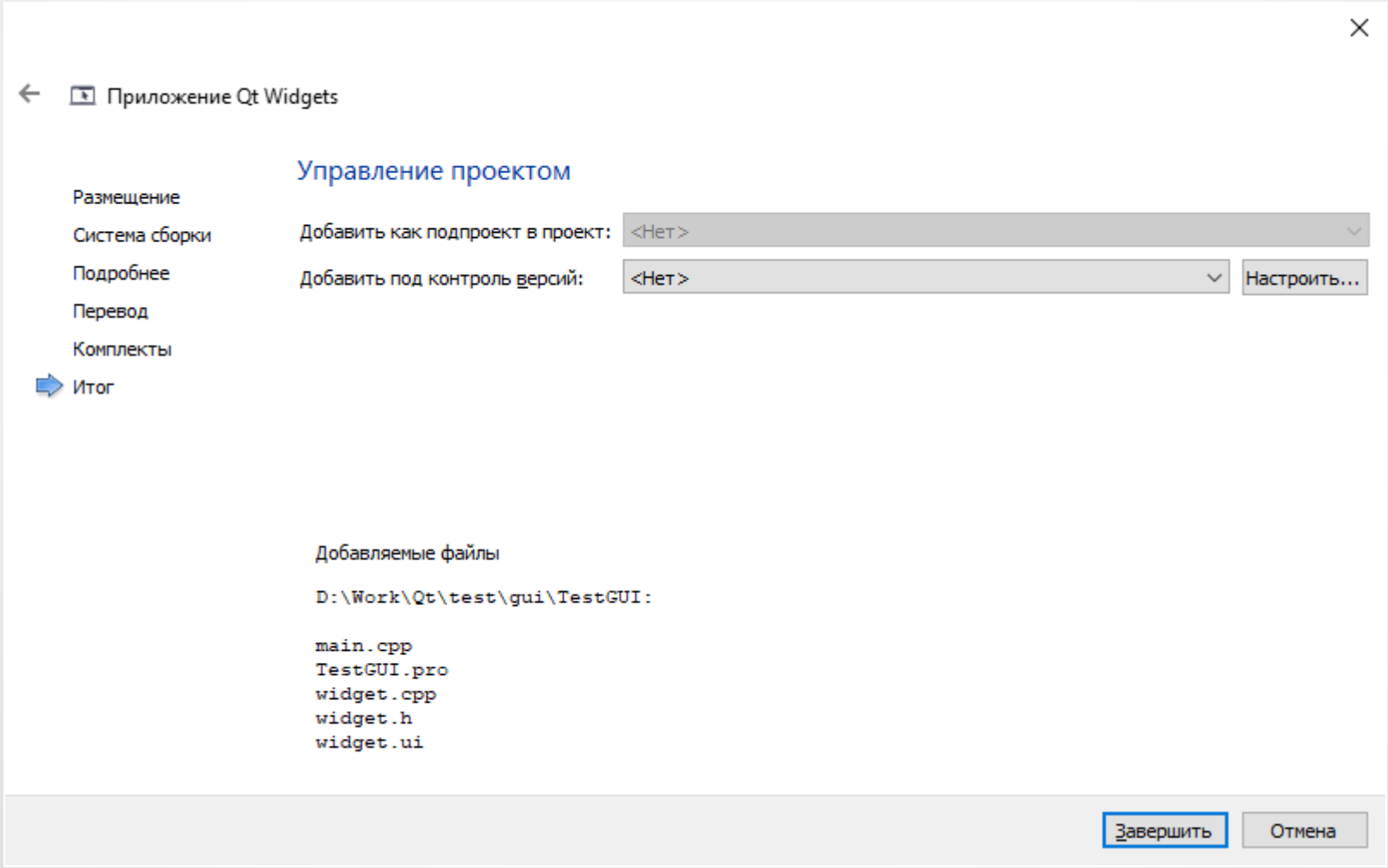
2. Простейшее приложение с GUI в Qt

7) выбрать комплект компилятора и отладчика (зависит от используемой ОС, установленных в ней программ и требований к приложению);



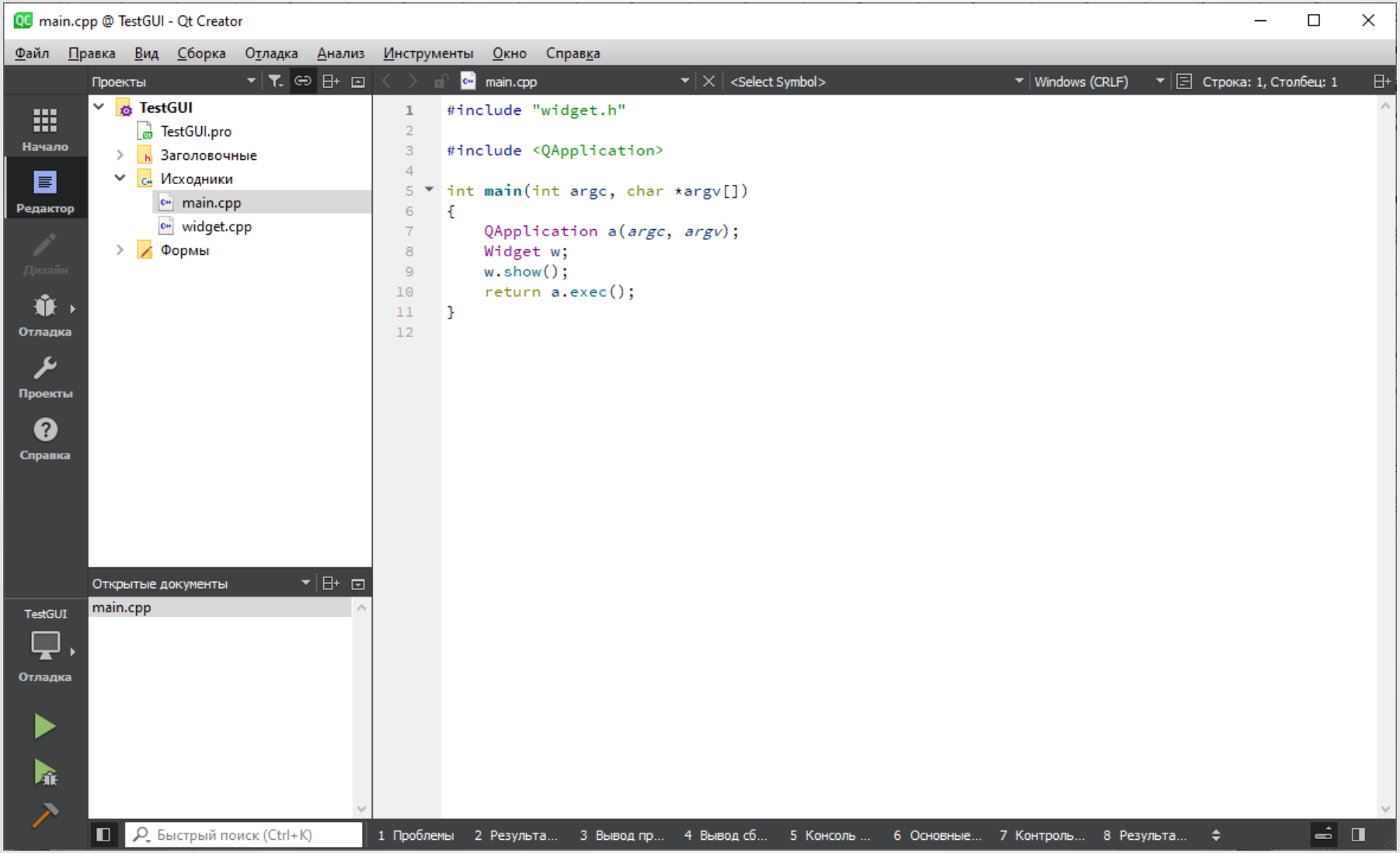
2. Простейшее приложение с GUI в Qt

8) выбрать способ управления проектом;



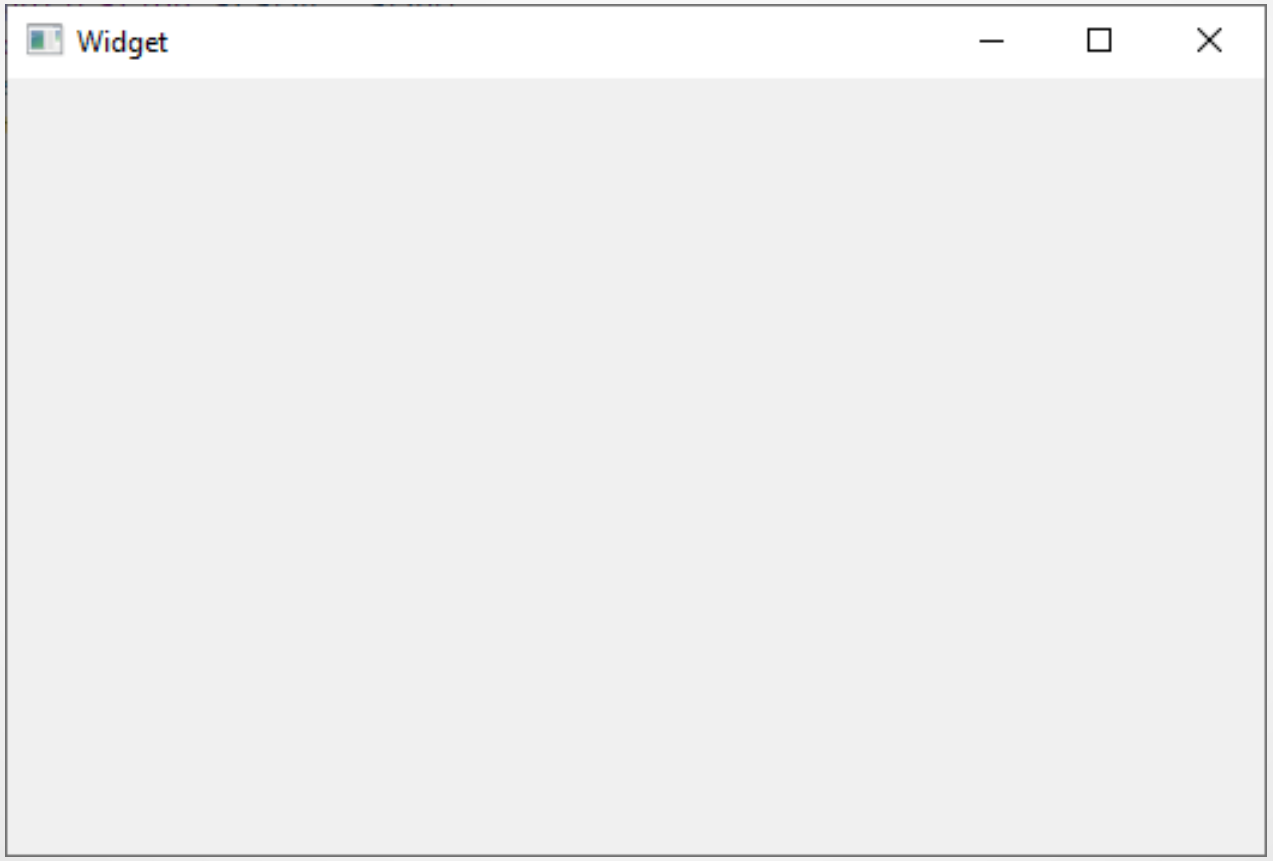
2. Простейшее приложение с GUI в Qt

После чего вновь созданный проект будет отображен в Qt Creator.



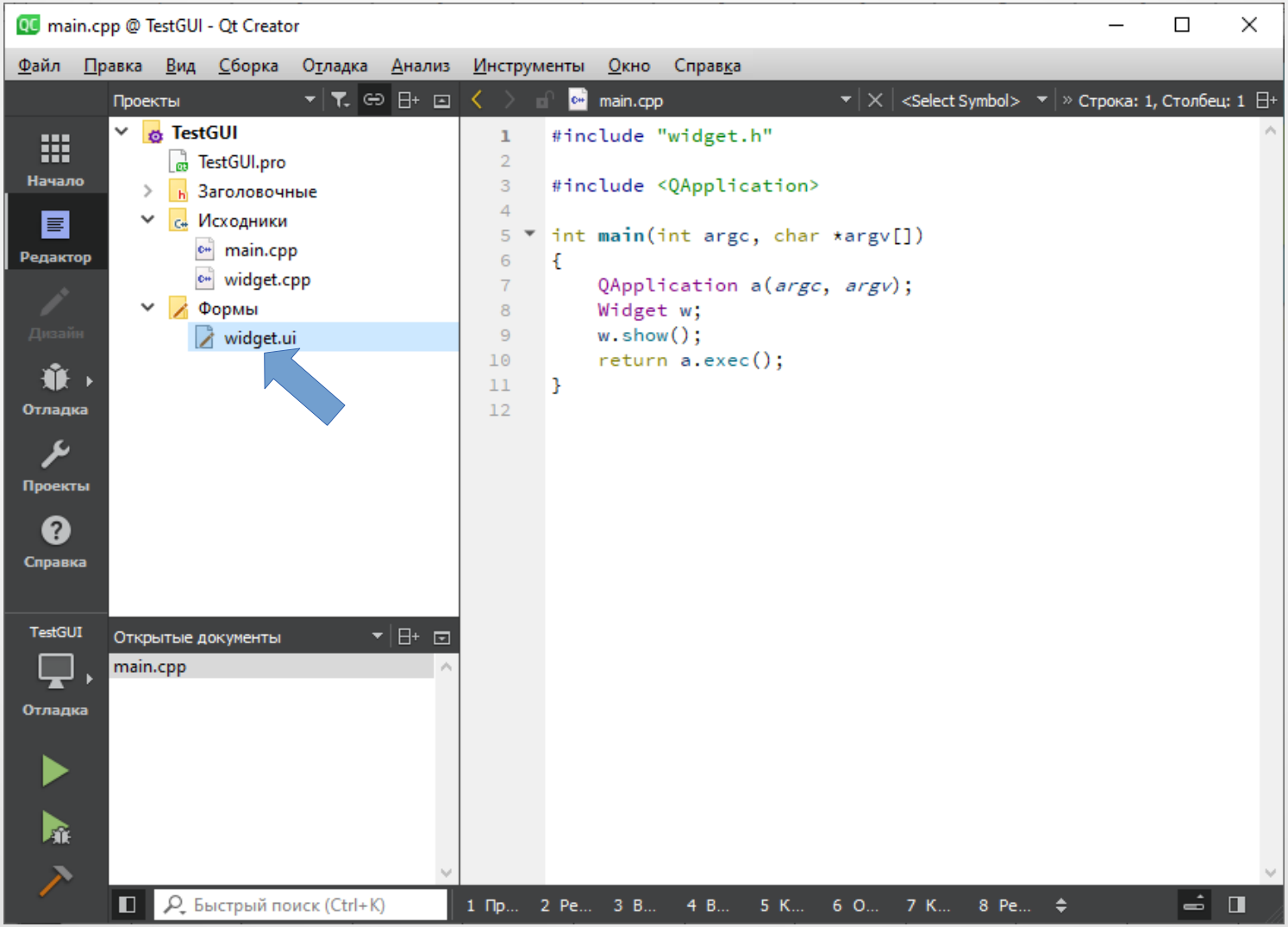
2. Простейшее приложение с GUI в Qt

Компиляция и запуск данной программы должны привести к следующему результату:



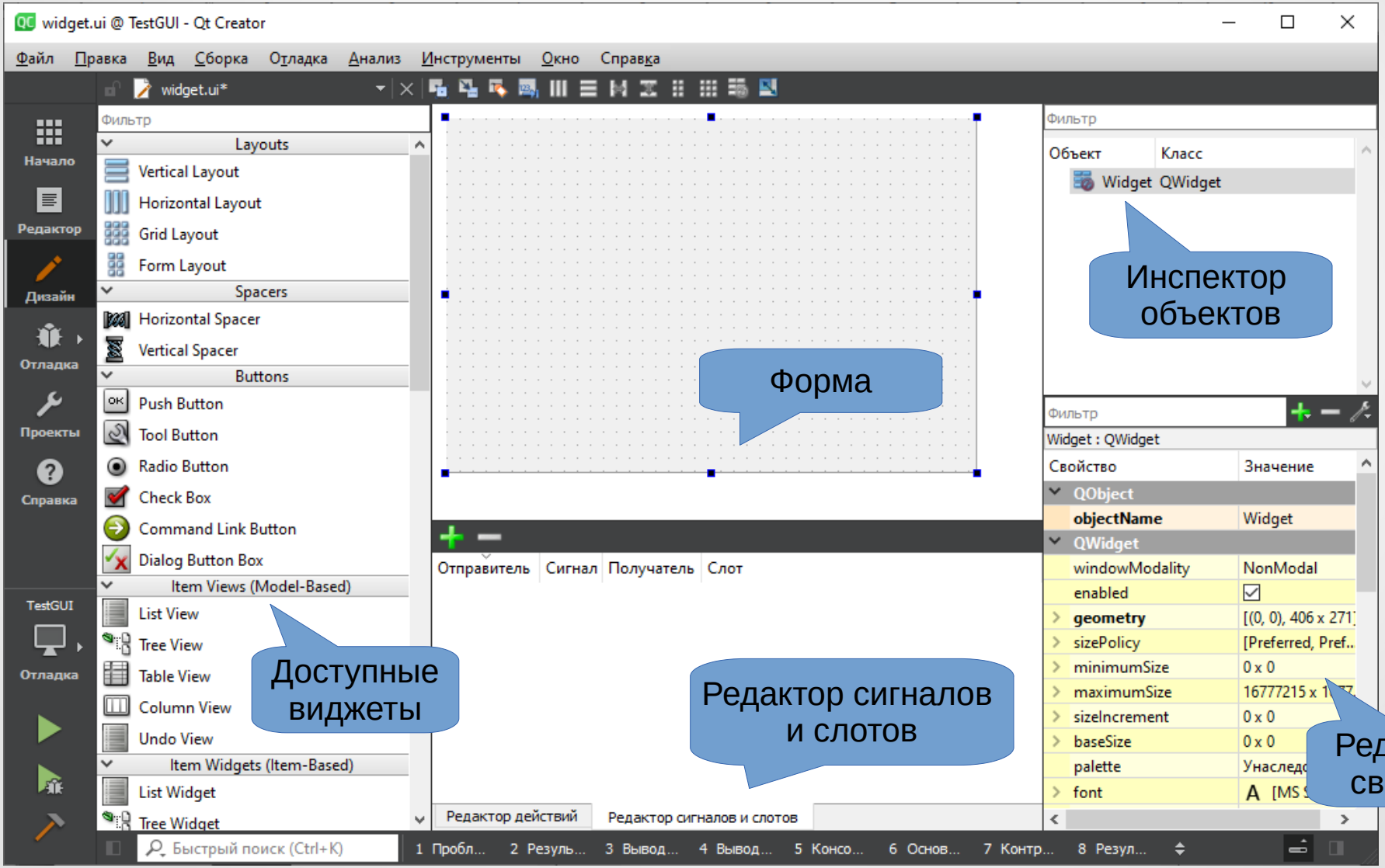
3. Редактирование формы в Qt Designer

Для редактирования формы ее предварительно нужно открыть (дважды кликнув в окне «Проекты» на соответствующей строчке):



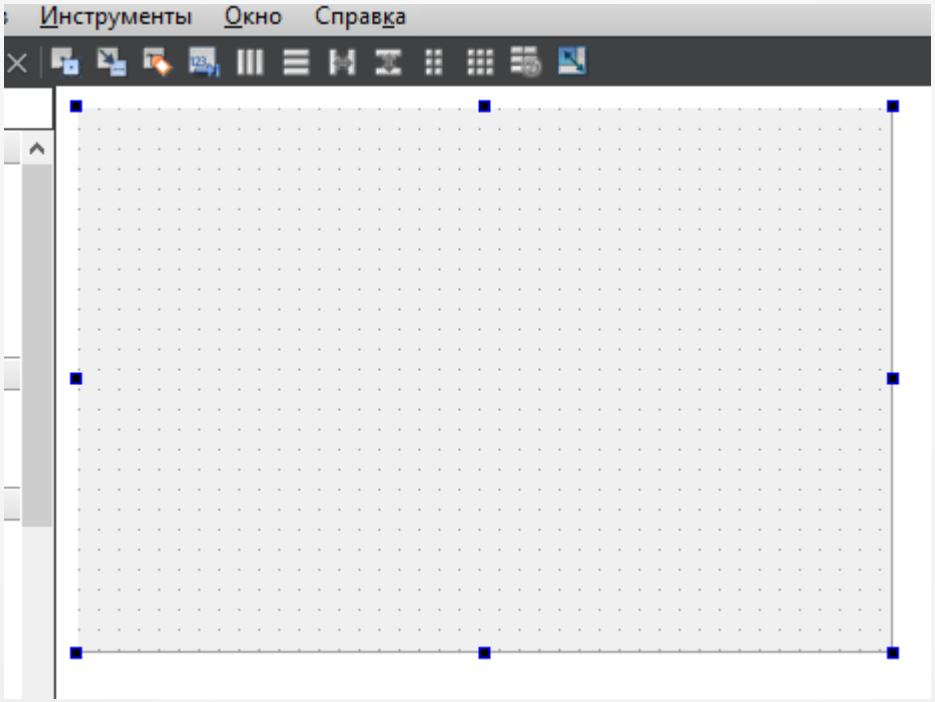
3. Редактирование формы в Qt Designer

После чего соответствующая форма будет открыта Qt Designer:



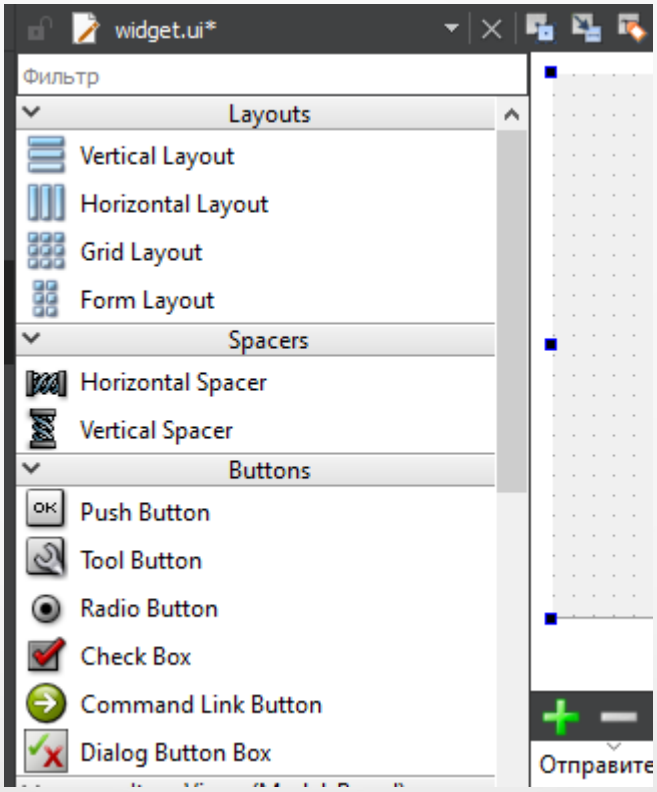
3. Редактирование формы в Qt Designer

Форма – центральный элемент дизайна программы. Это прототип будущего окна (главного или диалогового) приложения.



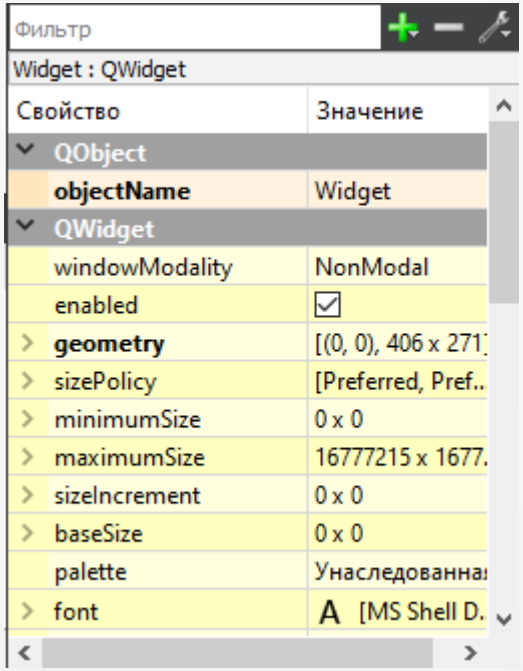
3. Редактирование формы в Qt Designer

Окно **доступных виджетов** (Widget Box) содержит список имеющихся в наличии виджетов и объектов компоновки, сгруппированных в отдельные категории. Перетаскиванием из этого окна на форму нужных пользователю компонентов осуществляется добавляются их к программе.



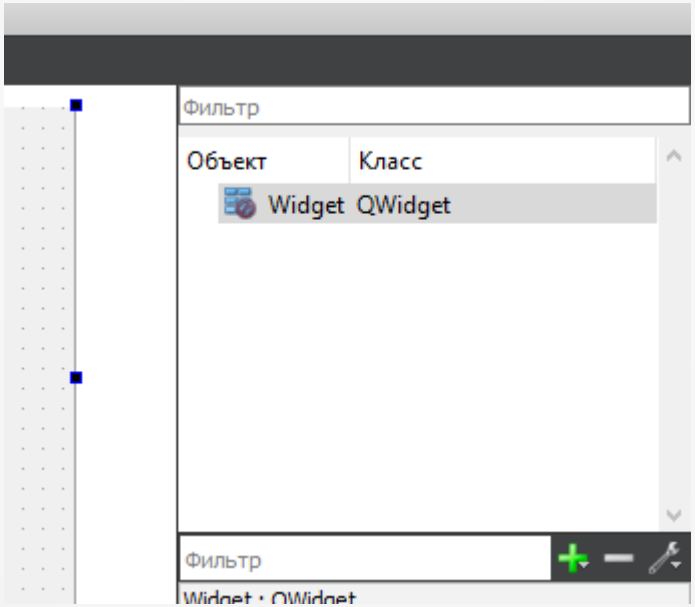
3. Редактирование формы в Qt Designer

Редактор свойств (Property Editor) – содержит базовые свойства текущего виджета. Например, это могут быть **цвет фона**, **шрифт**, **максимальный** и **минимальный размер виджета** и т. п.



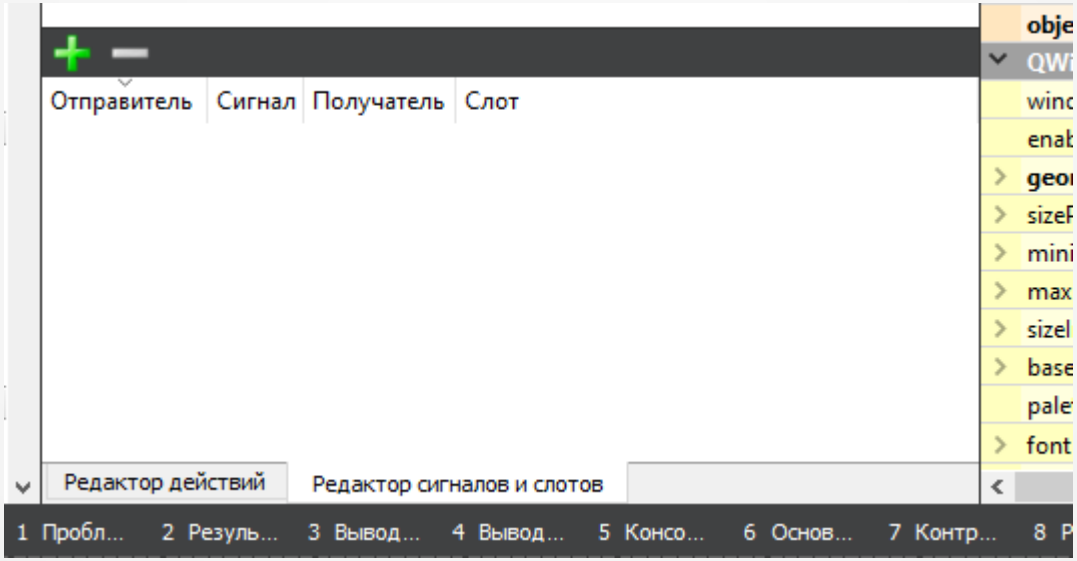
3. Редактирование формы в Qt Designer

Инспектор объектов (Object Inspector) – отображает список используемых виджетов. В этом окне их можно выбирать для последующего изменения.



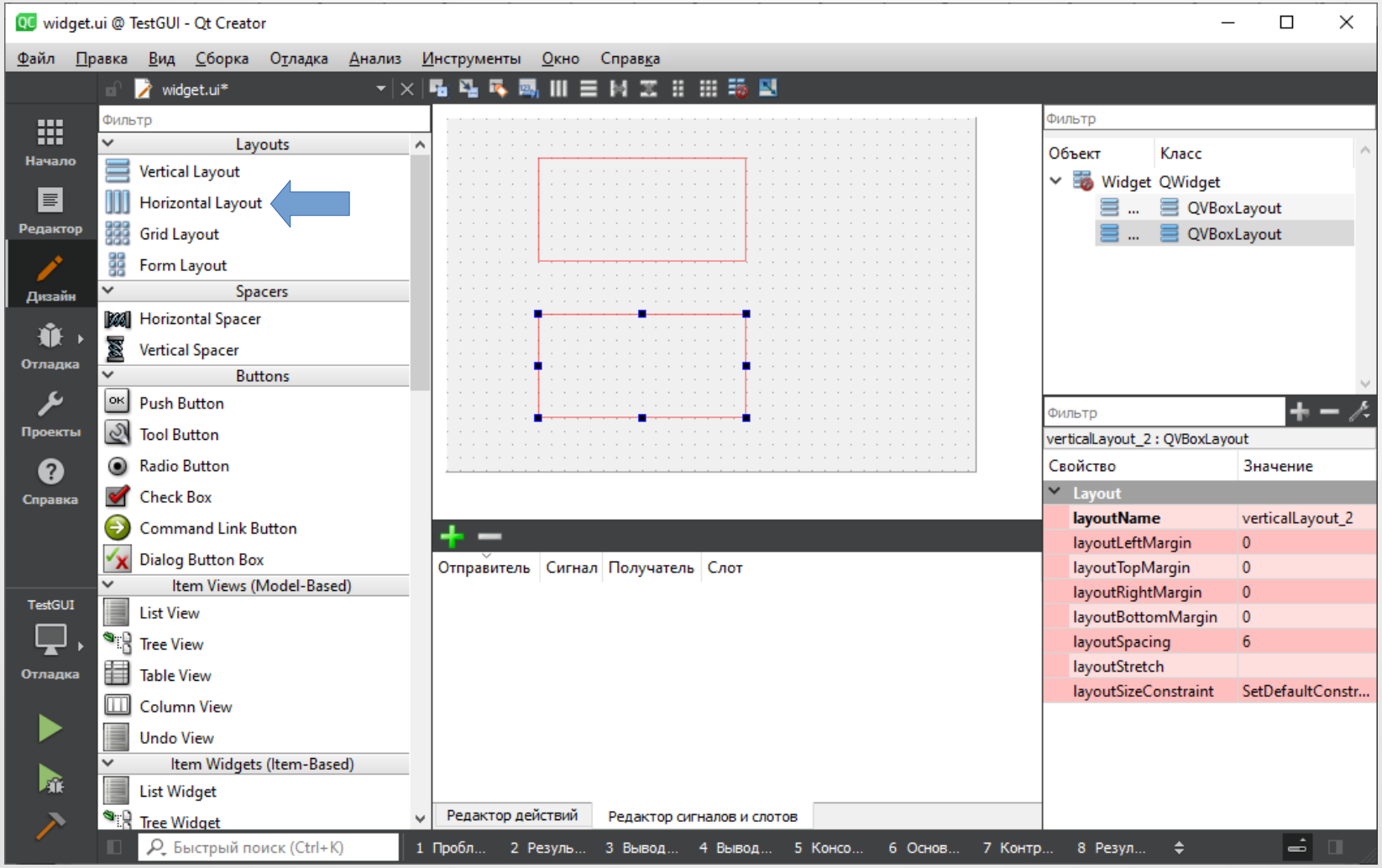
3. Редактирование формы в Qt Designer

Редактор сигналов и слотов (Signal/Slot Editor) – окно, предназначенное для редактирования соединений сигналов со слотами.



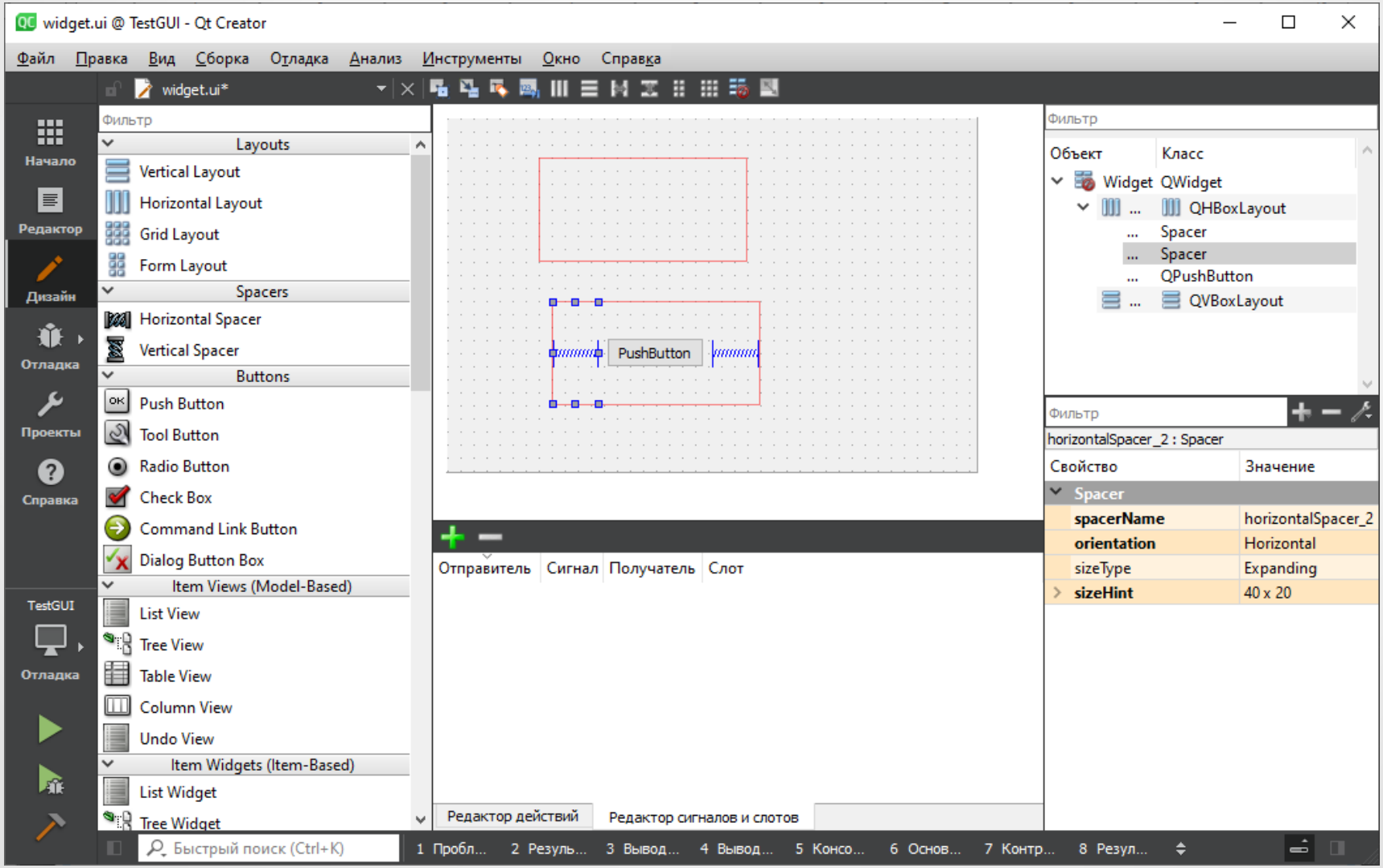
3. Редактирование формы в Qt Designer

Для добавления нового виджета на форму его нужно выбрать в окне доступных виджетов и перетянуть на форму. Например, перетянем на форму два горизонтальных компоновщика:



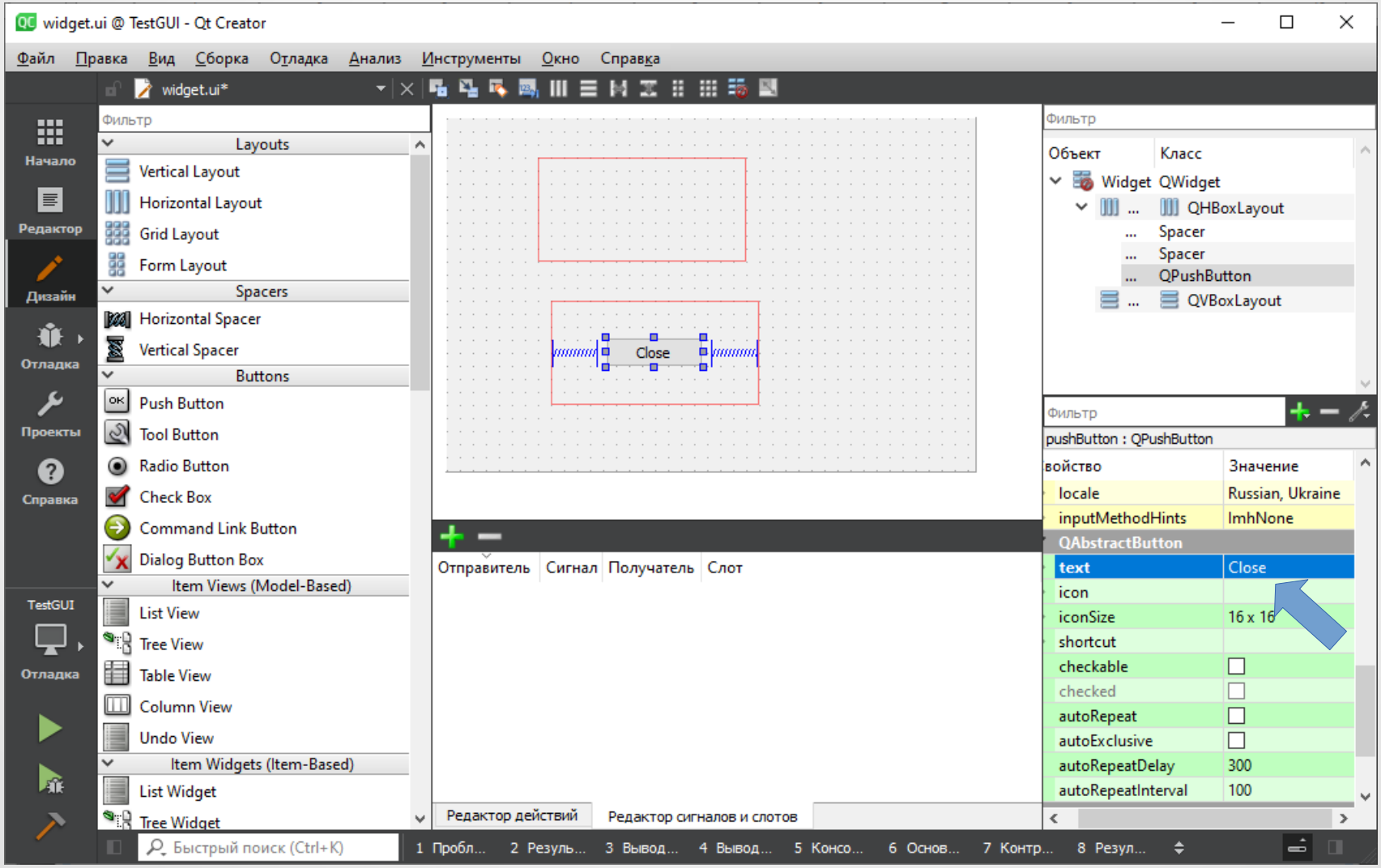
3. Редактирование формы в Qt Designer

В нижний компоновщик поместим кнопку (**Push Button**) и два горизонтальных заполнителя (**Horizontal Spacer**):



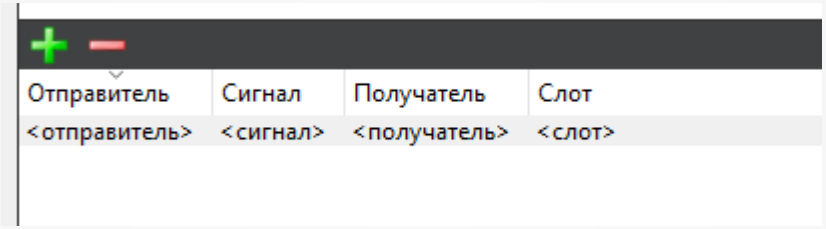
3. Редактирование формы в Qt Designer

Кликнув на кнопку, сделаем ее активной. В окне редактора свойств найдем свойство **Text** (надпись на кнопке) и заменим стандартный текст на «Close».

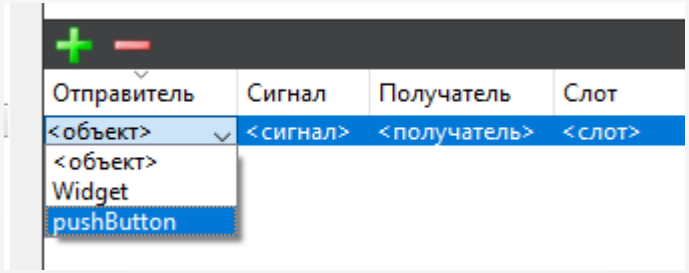


3. Редактирование формы в Qt Designer

Настроим данную кнопку таким образом, чтобы при ее нажатии форма (окно) закрывалась (**закрытие главного окна программы автоматически приводит к ее завершению**). Для этого в окне редактора сигналов и слотов нажмем на кнопку «+», после чего будет добавлена новая строчка настроек:

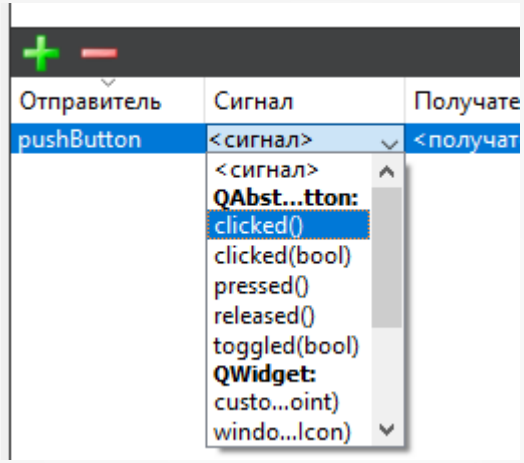


Изменим значение поля «Отправитель», выбрав из списка значение «pushButton»

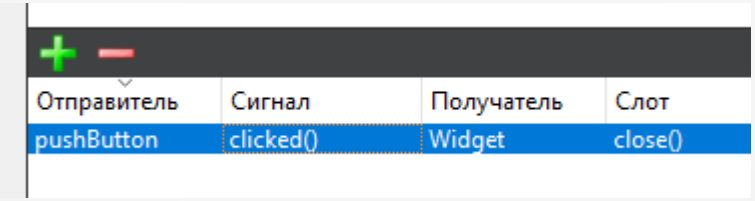


3. Редактирование формы в Qt Designer

Затем в поле «Сигнал» выберем значение «clicked()»:



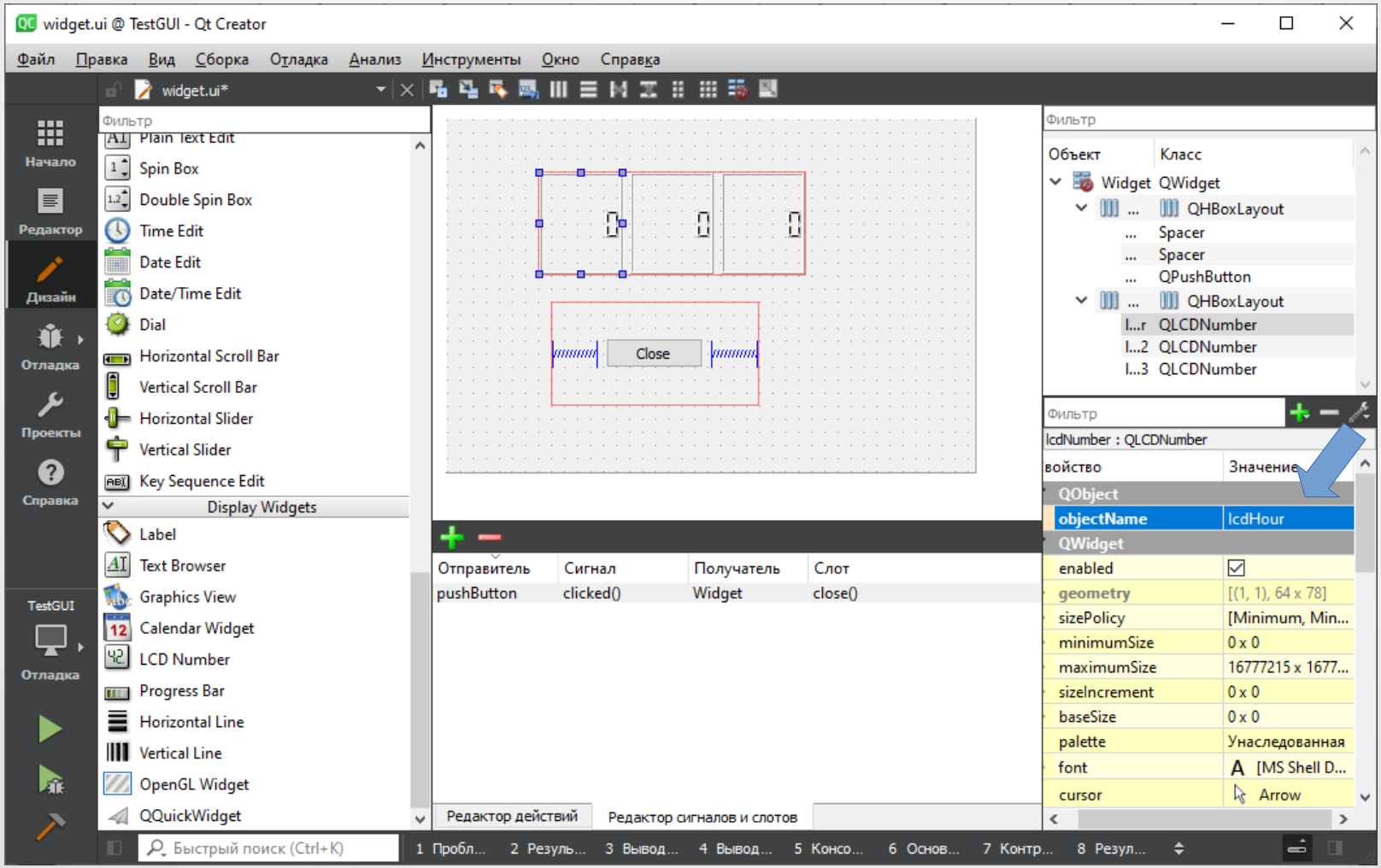
Аналогичным образом настроим поля «Получатель» и «Слот», выбрав для них значения «Widget» и «close()».



Таким образом, при нажатию на кнопку «pushButton» будет генерироваться сигнал «clicked()», который будет вызывать слот «close()» у формы «Widget». Что в конечном итоге будет приводить к закрытию окна.

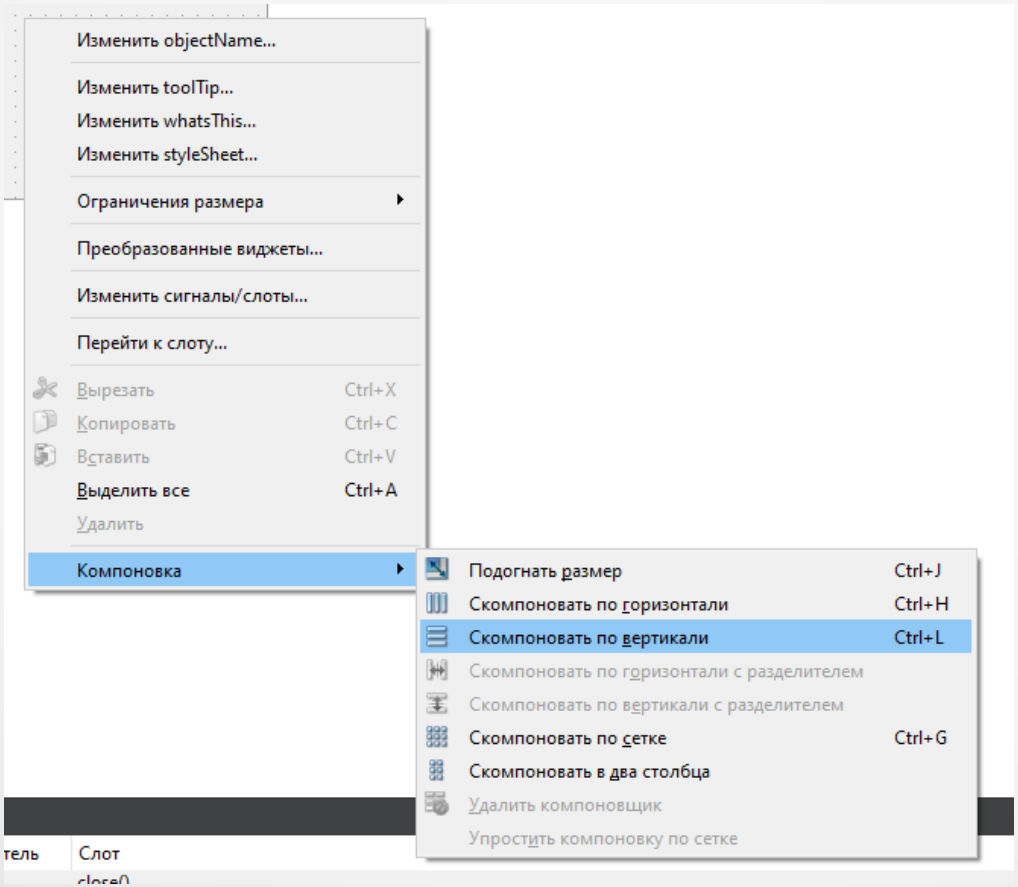
3. Редактирование формы в Qt Designer

В верхний вертикальный компоновщик добавим три виджета «LCD Number». С помощью свойства «**objectName**» в окне редактора свойств переименуем эти объекты в «lcdHour», «lcdMin» и «lcdSec»:



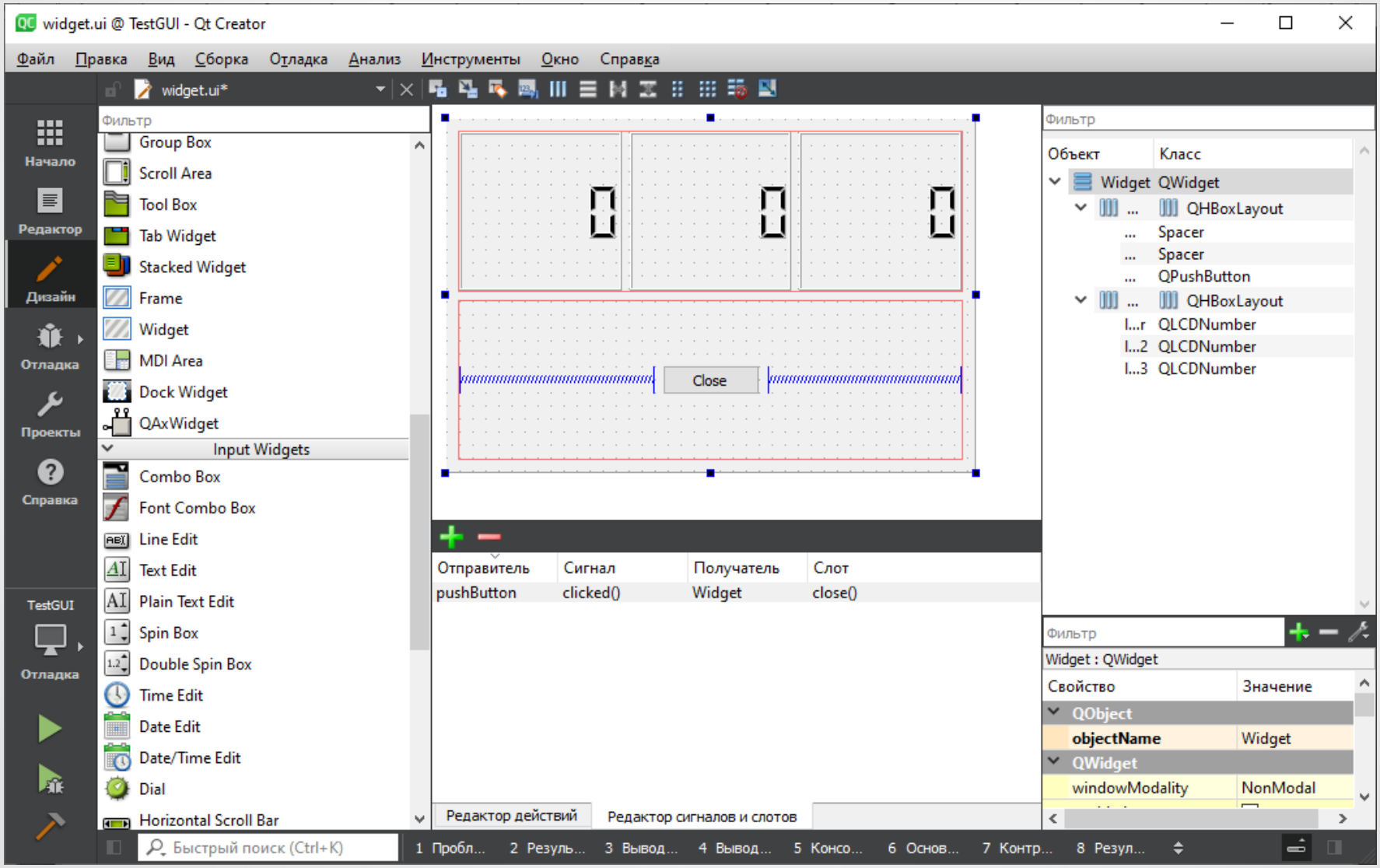
3. Редактирование формы в Qt Designer

Кликнем на форме **правой кнопкой** мышки и в появившемся контекстном меню выберем команду «**Компоновка | Скомпоновать по вертикали**»:



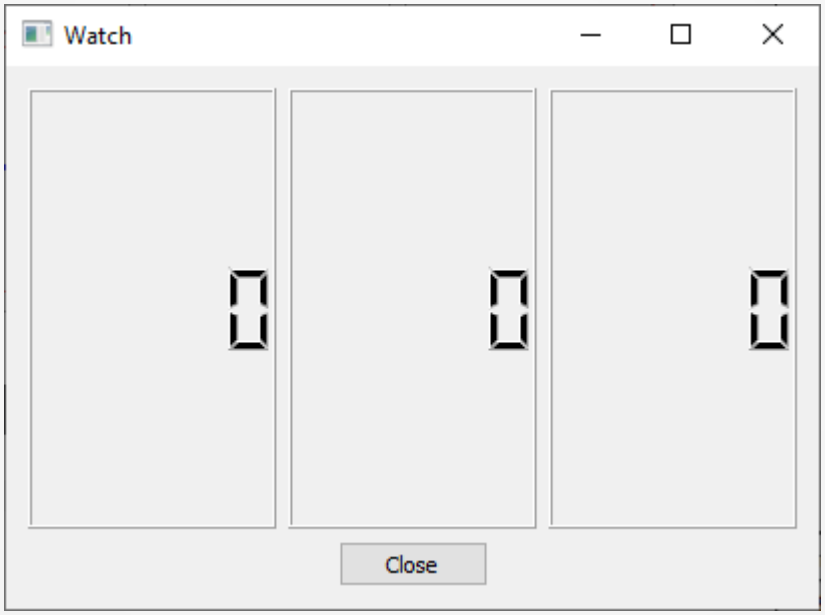
3. Редактирование формы в Qt Designer

После чего компоненты на форме выровняются таким образом:



3. Редактирование формы в Qt Designer

Изменим заголовок формы (свойство «windowTitle» в редакторе свойств) на «Watch» и запустим программу. Получим следующий результат:



3. Редактирование формы в Qt Designer

Для завершения разработки часов с цифровой индикацией времени осталось реализовать функционал изменения значения виджетов, отображающих часы, минуты и секунды соответствующими значениями текущего времени.

Заголовочный файл widget.h

```
#ifndef WIDGET_H
#define WIDGET_H

#include <QWidget>

QT_BEGIN_NAMESPACE
namespace Ui { class Widget; }
QT_END_NAMESPACE

class Widget : public QWidget
{
    Q_OBJECT
private:
    QTimer *timer;
private slots:
    // Изменение часов
    void sloteUpdateWatch(void);

public:
    Widget(QWidget *parent = nullptr);
    ~Widget();

private:
    Ui::Widget *ui;
};
#endif // WIDGET_H
```

3. Редактирование формы в Qt Designer

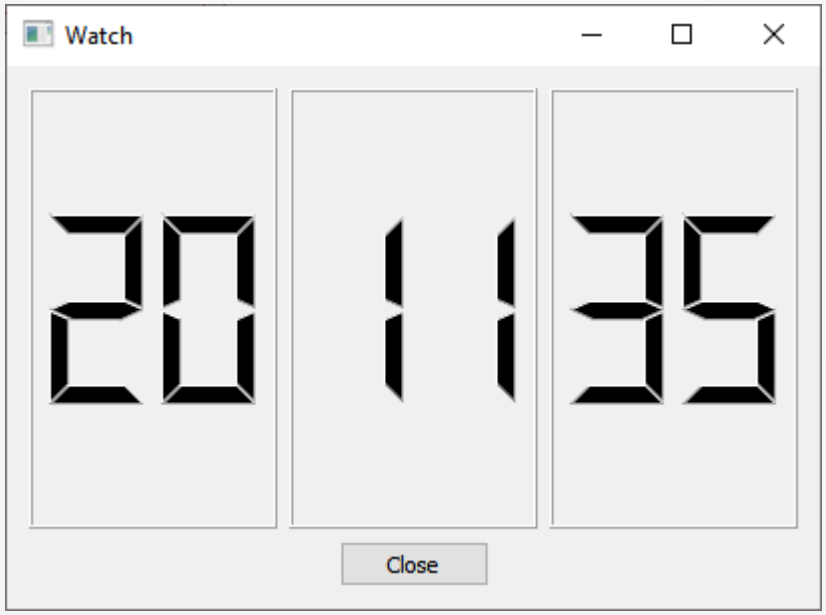
Файл widget.cpp

```
#include <QTime>
#include <QTimer>
#include "widget.h"
#include "ui_widget.h"

Widget::Widget(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::Widget)
{
    ui->setupUi(this);
    timer = new QTimer();
    connect(timer, SIGNAL(timeout()), this,
                SLOT(sloteUpdateWatch()));
    timer->start(1000);
}

Widget::~Widget()
{
    delete timer;
    delete ui;
}
```

```
void Widget::sloteUpdateWatch(void)
{
    QTime time = QTime::currentTime();

    ui->lcdHour->display(time.hour());
    ui->lcdMin->display(time.minute());
    ui->lcdSec->display(time.second());
}
```

3. Редактирование формы в Qt Designer

Компиляция и запуск данной программы приведут к следующему результату:



Примечание. В приведенном примере у объектов типа LCDNumber было изменено на «2» свойство «digitCount», содержащее количество отображаемых цифр.